

Sisteme de recunoaștere a formelor – Laborator 10

Clasificatori liniari și algoritmul perceptron

1. Obiective

Acest laborator prezintă algoritmul de învățare perceptron pentru clasificatori liniari. Vom aplica gradient descent și stochastic gradient descent pentru a obține vectorul de ponderi. Se va rezolva o problemă de clasificare binară folosind două trăsături.

2. Fundamente teoretice

Scopul clasificării este determinarea clasei unor exemple descrise prin trăsături. Clasa unui exemplu face parte dintr-o mulțime finită, de exemplu: roșu/albastru, pieton/fundal, o cifră din mulțimea 0, 1, ..., 9. Un clasificator liniar reușește acest lucru prin luarea deciziei în funcție de o combinație liniară a trăsăturilor.

Definiții

Definim un dataset etichetat ca și o pereche (X, Y) , unde $X \in M_{n \times m}(R)$ este o matrice cu n linii și m coloane având valori reale și Y este un vector coloană $Y \in M_{n \times 1}(D)$, care conține etichetele pentru fiecare instanță, iar D este mulțimea claselor posibile. X conține pe linii instanțele de antrenare descrise cu m trăsături. Un clasificator este o funcție care mapează orice instanță reprezentată printr-un vector de trăsături la o clasă: $f: R^m \rightarrow D$.

Astfel un clasificator separă spațiul de trăsături în $|D|$ regiuni disjuncte (numărul de elemente din D). Supspațiul care separă clasele se numește suprafața de decizie (eng. decision boundary). Dacă problema de clasificare este bidimensională acest subspațiu va fi unidimensional și va fi o linie sau o curbă, în general. În continuare vom discuta despre cazul cu două clase, algoritmi fiind ușor extensibili la mai multe clase. Pentru etichete vom folosi $D = \{-1, 1\}$.

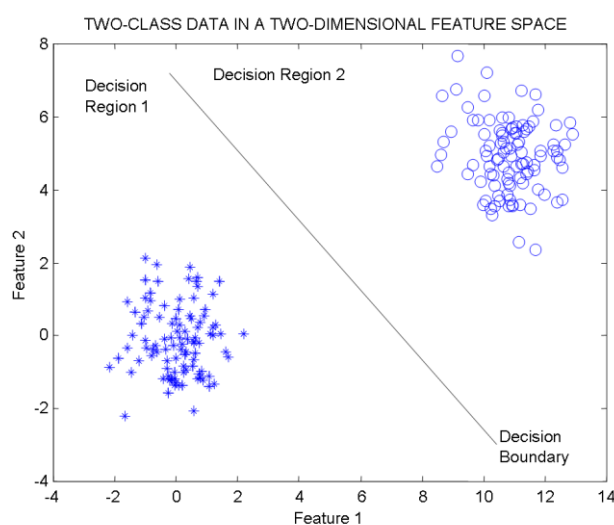


Figure 1. Exemplu de clasificator liniar care separă două clase și utilizează două trăsături

2.1. Forma generală a unui clasificator liniar

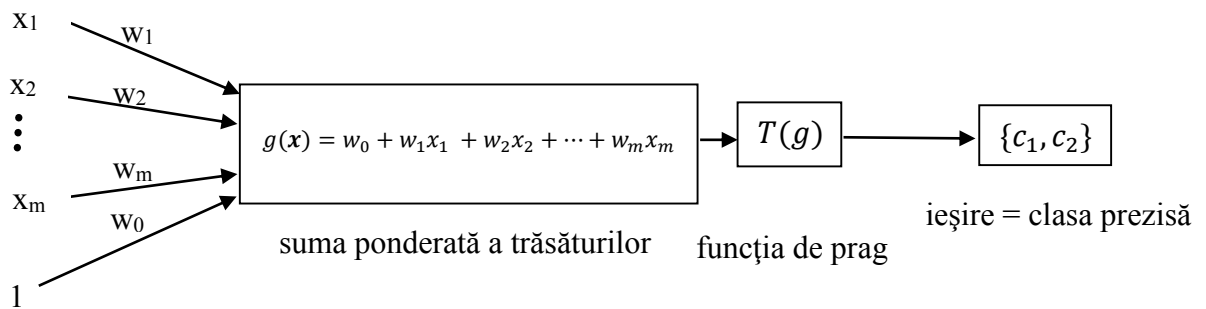
Un clasificator liniar este o combinație liniară a trăsăturilor de intrare. Dacă notăm cu $\mathbf{x} \in M_{1 \times m}(R)$ vectorul de trăsături și cu $\mathbf{w} \in M_{1 \times m}$ putem exprima funcția de clasificare ca și:

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^m w_i x_i = w_0 + \mathbf{w}\mathbf{x}^T$$

Unde

- \mathbf{w} este vectorul de ponderi
- w_0 este deplasamentul sau valoarea de prag.

Următoarea schemă ilustrează modul de operare a clasificatorului liniar:



Pentru a simplifica notația, calculele și pentru a permite tratarea generală vom absorbi valoarea w_0 în vectorul de ponderi și vom augmenta vectorul de trăsături cu o componentă egală cu 1. În acest caz se simplifică expresia în:

$$w_0 + \mathbf{w}\mathbf{x}^T = [w_0 \quad \mathbf{w}] \begin{bmatrix} 1 \\ \mathbf{x}^T \end{bmatrix} = \bar{\mathbf{w}}\bar{\mathbf{x}}^T$$

Un clasificator liniar pentru două clase implementează următoarea regulă de clasificare:

- dacă $g(\mathbf{x}) > 0$, decidem că \mathbf{x} aparține clasei +1;
- dacă $g(\mathbf{x}) < 0$, decidem că \mathbf{x} aparține clasei -1;

sau echivalent:

- dacă $\mathbf{w}\mathbf{x}^T > -w_0$, decidem că \mathbf{x} aparține clasei +1;
- dacă $\mathbf{w}\mathbf{x}^T < -w_0$, decidem că \mathbf{x} aparține clasei -1;

Dacă $g(\mathbf{x}) = 0$, atunci \mathbf{x} poate fi atribuit la oricare clasă, fiind pe suprafața de separare.

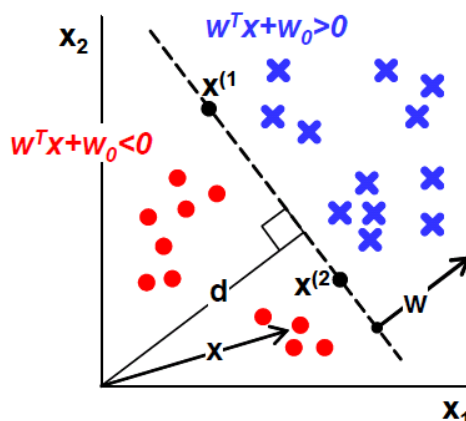


Figure 2. Ilustrarea componentelor din vectorul de ponderi: vectorul w este normala la dreapta de separare iar w_0 este egal cu distanța d (cu semn) a dreptei de la origine înmulțită cu $\|w\|$ norma lui w

2.2. Metode de învățare pentru clasificatori liniari

Vom prezenta două metode de învățare pentru clasificatori liniari. Învățarea se face prin transformarea problemei de clasificare într-o problemă de minimizare a unei funcții de cost. Denumim funcția L (de la loss function din engleză). Funcția poate să aibă mai multe forme și trebuie să penalizeze diferențele dintre predicția clasificatorului și clasa adevărată. În cazul algoritmului perceptron adoptăm următoarea funcție de cost:

$$L(\bar{w}) = \frac{1}{n} \sum_{i=1}^n \max(0, -y_i \bar{w} \cdot \bar{x}_i^T) = \frac{1}{n} \sum_{i=1}^n L_i(\bar{w})$$

Dacă o instanță este corect clasificată atunci nu se aplică nici o penalizare, în schimb, dacă instanța se atribuie la clasa greșită atunci se penalizează proporțional cu scorul greșit. Semnul expresiei $y_i \bar{w}^T \cdot \bar{x}_i$ este negativ dacă semnul etichetei de clasă y_i este diferit de semnul clasei prezise $g(x)$. Funcția de cost se mai poate scrie și ca:

$$L(\bar{w}) = \frac{1}{n} \sum_{i \text{ clasificat greșit}} -y_i \bar{w} \cdot \bar{x}_i^T$$

Pentru a minimiza acest cost vom utiliza metoda gradient descent. Vom porni de la un vector de ponderi aleator și vom schimba acest vector bazându-ne pe valoarea gradientului în poziția curentă. Pasul va fi în direcția opusă a gradientului:

$$\bar{w}_{k+1} \leftarrow \bar{w}_k - \eta \nabla L(\bar{w}_k)$$

unde \bar{w}_k este vectorul de ponderi la momentul k , parametrul η controlează mărimea pasului și se numește rata de învățare (eng. learning rate), și $\nabla L(\bar{w}_k)$ este vectorul de gradient calculat în punctul \bar{w}_k . Vectorul de gradient are expresia:

$$\nabla L(\bar{w}) = \frac{1}{n} \sum_{i=1}^n \nabla L_i(\bar{w})$$

$$\nabla L_i(\bar{\mathbf{w}}) = \begin{cases} 0, & \text{daca } y_i \bar{\mathbf{w}} \cdot \bar{\mathbf{x}}_i^T > 0 \\ -y_i \bar{\mathbf{x}}_i, & \text{altfel} \end{cases}$$

În metoda care folosește gradientul pe setul întreg ponderile se schimbă doar după ce am vizitat toate exemplele de antrenare. Această abordare se mai cheama și batch-update. Algoritmul de perceptron din lucrarea [1] actualizează ponderile după examinarea fiecărui exemplu de antrenare. În acest caz algoritmul se numește online perceptron. Regula de actualizare devine:

$$\bar{\mathbf{w}}_{k+1} \leftarrow \bar{\mathbf{w}}_k - \eta \nabla L_i(\bar{\mathbf{w}})$$

Redăm în continuare, în paralel, cele două variante ale algoritmului de învățare:

Algorithm:

Batch Perceptron

```

init  $\mathbf{w}$ ,  $\eta$ ,  $E_{limit}$ , max_iter
for iter=1:max_iter
    E = 0, L = 0
     $\nabla L$  = [0,0,0]
    for i=1:n
         $z_i = \sum_{j=0}^m w_j X_{ij}$ 
        if  $z_i \cdot y_i \leq 0$ 
             $\nabla L \leftarrow \nabla L - y_i \mathbf{X}_i$ 
            E  $\leftarrow$  E + 1
            L  $\leftarrow$  L -  $y_i z_i$ 
        endif
    endfor
    E  $\leftarrow$  E/n
    L  $\leftarrow$  L/n
     $\nabla L \leftarrow \nabla L / n$ 
    if E <  $E_{limit}$ 
        break
     $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla L$ 
endfor

```

Algorithm:

Online Perceptron

```

init  $\mathbf{w}$ ,  $\eta$ ,  $E_{limit}$ , max_iter
for iter=1:max_iter
    E = 0
    for i=1:n
         $z_i = \sum_{j=0}^m w_j X_{ij}$ 
        if  $z_i \cdot y_i \leq 0$ 
             $\mathbf{w} \leftarrow \mathbf{w} + \eta \mathbf{X}_i y_i$ 
            E  $\leftarrow$  E + 1
        endif
    endfor
    E  $\leftarrow$  E/n
    if E <  $E_{limit}$ 
        break
    endfor

```

3. Detalii de implementare

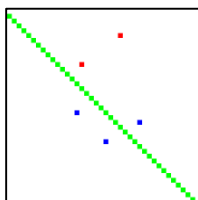
Vom folosi puncte 2D care sunt împărțite în 2 clase. Punctele se citesc din imaginile de intrare și apartenența lor se stabilește pe baza culorii. Asociem clasa -1 la punctele albastre și 1 la punctele roșii.

Fiecare punct va fi descris de două trăsături x_1 și x_2 care coincid cu coordonatele lor x și y, respectiv. Astfel, vectorul de trăsături augmentat are forma $\bar{\mathbf{x}} = [1 \ x_1 \ x_2]$ iar vectorul de ponderi are trei componente $\bar{\mathbf{w}} = [w_0 \ w_1 \ w_2]$.

4. Exemplu numeric

Fie punctele din fișierul points00 – (x,y) sau (coloana, rand):

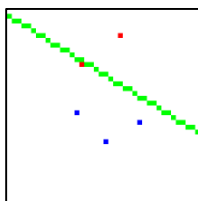
- Rosii: (23, 5), (15,11) – clasa +1
- Albastre: (14, 21), (27,23), (20, 27) – clasa -1



Rata de învățare = 0.01

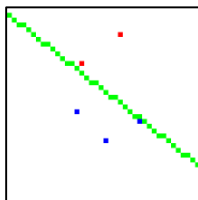
Iterația 0

```
i=0: w=[1.000000 1.000000 -1.000000] xi=[1 23 5] yi = 1 zi=19.000000
i=1: w=[1.000000 1.000000 -1.000000] xi=[1 15 11] yi = 1 zi=5.000000
i=2: w=[1.000000 1.000000 -1.000000] xi=[1 14 21] yi = -1 zi=-6.000000
i=3: w=[1.000000 1.000000 -1.000000] xi=[1 27 23] yi = -1 zi=5.000000 greșit
      update w0 = w0 - 0.01, w1 = w1 - 27*0.01, w2 = w2 - 23*0.01
i=4: w=[0.990000 0.730000 -1.230000] xi=[1 20 27] yi = -1 zi=-17.620000
```



Iterația 1

```
i=0: w= [0.990000 0.730000 -1.230000] xi= [1 23 5] yi = 1 zi=11.630000
i=1: w= [0.990000 0.730000 -1.230000] xi= [1 15 11] yi = 1 zi=-1.590000 greșit
      update w0 = w0 + 0.01, w1 = w1 + 15*0.01, w2 = w2 + 11*0.01
i=2: w= [1.000000 0.880000 -1.120000] xi= [1 14 21] yi = -1 zi=-10.200000
i=3: w= [1.000000 0.880000 -1.120000] xi= [1 27 23] yi = -1 zi=-1.000000
i=4: w= [1.000000 0.880000 -1.120000] xi= [1 20 27] yi = -1 zi=-11.640000
```



Iterația 2

```
i=0: w= [1.000000 0.880000 -1.120000] xi= [1 23 5] yi = 1 zi=15.640000
i=1: w= [1.000000 0.880000 -1.120000] xi= [1 15 11] yi = 1 zi=1.880000
i=2: w= [1.000000 0.880000 -1.120000] xi= [1 14 21] yi = -1 zi=-10.200000
i=3: w= [1.000000 0.880000 -1.120000] xi= [1 27 23] yi = -1 zi=-1.000000
i=4: w= [1.000000 0.880000 -1.120000] xi= [1 20 27] yi = -1 zi=-11.640000
```

Toate clasificate corect

5. Lucrare practică

1. Citiți punctele dintr-o singură imagine *test0*.bmp* și construiți setul de antrenare (X, Y) . Asociați -1 la punctele albastre și +1 la punctele roșii.
2. Implementați algoritmul *Online Perceptron* pentru a găsi linia de separare dintre cele două clase. Sugestie pentru parametri:
 $\eta = 10^{-4}$, $w_0 = [0.1, 0.1, 0.1]$, $E_{limit} = 10^{-5}$, $max_iter = 10^5$.
Observație: Pentru a accelera convergența algoritmului se folosește o rata de învățare mai mare doar pentru termenul liber w_0 .
3. Desenați linia de separare găsită de algoritm și dată de ecuația:
$$w_0 + w_1x + w_2y = 0.$$
4. Implementați algoritmul de batch perceptron. Găsiți parametri buni care asigură învățarea. Se va monitoriza funcția de cost care trebuie să descrească încet la fiecare pas.
5. Vizualizați linia de separare în timp ce rulează algoritmul să observați cum se schimbă.
6. Schimbați valorile de start pentru vectorul \mathbf{w} , modificați rata de învățare și observați ce se întâmplă în fiecare caz. Ce înseamnă dacă funcția de cost L oscilează în loc să descrească la fiecare pas?

6. Referințe

- [1] Rosenblatt, Frank (1957), The Perceptron - a perceiving and recognizing automaton. Report 85-460-1, Cornell Aeronautical Laboratory.
- [2] Richard O. Duda, Peter E. Hart, David G. Stork: Pattern Classification 2nd ed.
- [3] Xiaoli Z. Fern, Machine Learning Course, Oregon University - <http://web.engr.oregonstate.edu/~xfern/classes/cs534/notes/perceptron-4-11.pdf>
- [4] Gradient Descent - http://en.wikipedia.org/wiki/Gradient_descent
- [5] Avrim Blum, Machine Learning Theory, Carnegie Mellon University - <https://www.cs.cmu.edu/~avrim/ML10/lect0125.pdf>